MILES BERRY PRINCIPAL LECTURER

TEACHERS' KNOWLEDGE

The three overlapping areas of knowledge explained

It's undoubtedly true that the quality of any education system is dependent on the quality of its teachers, and that the only way to get excellent computer science teaching in every school is to get an excellent teacher of computer science in every school. But what should an excellent teacher of computer science know? And what sorts of initial teacher training, or subsequent professional development, is likely to produce excellent teachers of computer science?

Back in 2006, Mischra and Koehler, building on earlier work by Shulman, developed the 'TPACK' framework: I think this provides one helpful way of attempting to address these questions. Put simply, TPACK recognises that there are three overlapping areas of knowledge which teachers need: technological knowledge, pedagogical knowledge, and content knowledge. Mishra and Koehler go into more detail in considering the interactions and overlaps between these three areas, but let's just think about the three areas in turn here.

Content

I'm at risk of stating the obvious, but in order to teach computer science well, a teacher has to know some computer science. Whilst we might argue that the role of the teacher is now much more 'guide on the side' than 'sage on the stage', there seems to be fairly robust evidence that pupils learn most when their teacher does know what they're talking about. In developing the 'Barefoot Computing' CPD for primary teachers, we started from the premise that teachers already knew how to teach and, by and large, knew how to use technology, but most in primary schools didn't know much, or any, computer science. Similarly the QuickStart guides I wrote for CAS/BCS were focussed quite tightly on giving teachers the subject knowledge they needed to teach primary and lower-secondary computing. There are many excellent resources out there for teachers who want to learn computer science: books are great, as are online interactive tutorials (to a point), as well as many MOOCs, including those from the Raspberry Pi Foundation.

Technology

Whilst many teachers do have a good level of technology skills relevant for their broader professional role, I'm not sure the same is true when it comes to the specifics of programming and digital making. Even coding in Scratch seems a big step up from creating presentations, and text-based coding, robotics, or data science make even bigger demands on teachers' technical skills. Classroom confidence does seem to depend, at least in part, on being a master of the tools of ones trade and, for the computer science teacher or digital making educator, these tools are typically quite complex, flexible, and powerful. Papert's great insight, that learning happens through making, applies to teachers as much as to their pupils: mastering the tools of our trade comes through using them productively, creatively and (often) collaboratively: sign up for the next Picademy to see this in action.

Pedagogy

How we teach computing is interesting territory, and I'm not sure that we can take this for granted. Most subjects in school have a long-established body of pedagogic practice: we teach, very often, how we were taught. This doesn't work for computer science: it's such a different subject from ICT that we can't assume what worked there works here, nor can those of us with a degree in CS simply take the practice from higher education and apply it in school. For me, this is what makes the subject so exciting: that teachers are by and large, figuring out for themselves what works: generously sharing their insights with their peers through CAS hubs or CSTA chapters, as well as experimenting with innovative approaches and writing up their findings, as in the BCS Certificate scheme.

All of this, of course, takes time. Great as Barefoot, Picademy, the BCS Certificate and all the other initiatives are, none will work if teachers don't have the time to participate in these, integrate these different forms of knowledge, and try what they've been taught out for themselves. (HW)

REFERENCES

Mishra, P. and Koehler, M.J., 2006. Technological pedagogical content knowledge: A framework for teacher knowledge. Teachers college record, 108(6), p.1017.

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. Educational Researcher, 15(2), 4-14

Miles is principal lecturer in computing education at the University of Roehampton. He is a member of the CAS and CSTA boards and the Raspberry Pi Foundation.